UNIVERSITÄT
BERN

TA Session 5

Anthony Gillioz
Institute of Computer Science – University of Bern, Switzerland

Contact: anthony.gillioz@unibe.ch

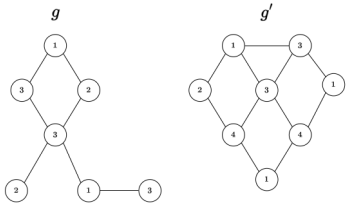
UNIVERSITÄT BERN

Theoretical Tasks

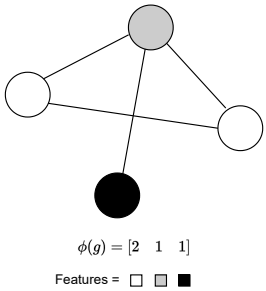
Task 1

1. Given the following graphs g and g', compute the angle $\measuredangle(g, g')$ between them by means of a *node feature kernel*.

   A node feature kernel is a type of kernel function that computes the graph similarity based on a their *feature vectors*. A feature vector is a vector of numerical values that describes the characteristics of the nodes. Each element in the vector corresponds to a specific feature or attribute of the node, such as the number of occurrences of one specific label.



- Compute the angle between g and g' by means of *node feature kernel*.



UNIVERSITÄT BERN

Theoretical Tasks

Task 2

2. Discuss the major benefits of the kernel trick and kernel machines for graph-based pattern recognition.

- Identify at least two points.

UNIVERSITÄT BERN

Theoretical Tasks

Task 3

3. Given the following dissimilarity matrix $\mathbf{D}$, compute the kernel matrix of the von Neumann diffusion kernel $\mathbf{K}$ (with $\lambda = 0.1$). Determine the value of t at which the sum of the diffusion kernel matrices converges (the step t where the difference between two consecutive matrices in the infinite sum is less than or equal to $\epsilon = 10^{-3}$ (i.e., $||\mathbf{M}_t - \mathbf{M}_{t-1}||_2 \leq \epsilon$)).

$$\mathbf{D} = \begin{bmatrix} 0 & 5 & 8 & 9 & 6 \\ 5 & 0 & 3 & 7 & 7 \\ 8 & 3 & 0 & 4 & 6 \\ 9 & 7 & 4 & 0 & 1 \\ 6 & 7 & 6 & 1 & 0 \end{bmatrix}$$

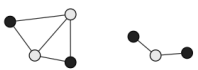
- Use the *von Neumann diffusion kernel* presented in the lecture notes.
- You can use Python for this exercise.
- Report M_t and t.

UNIVERSITÄT BERN

Theoretical Tasks

Task 4

4. Compute and illustrate the direct product graph for the following two graphs and the adjacency matrices $\mathbf{A}_{\times}^{n}$ with $n = \{1, 2, 3\}$ (the nodes are labeled with a binary label 'black' or 'gray'). Illustrate the meaning of an entry $a_{ij} = 4$ in $\mathbf{A}_{\times}^{2}$, and an entry $a_{ij} = 8$ in $\mathbf{A}_{\times}^{3}$.



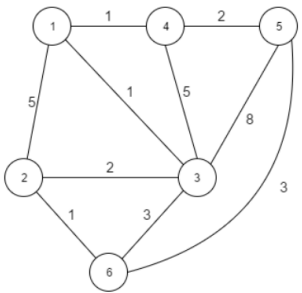
- Draw the direct product graph
- Write down the various adjacency matrices $\mathbf{A}_{\times}^{n}$ with $n = \{1, 2, 3\}$
- Explain what the entries in $\mathbf{A}_{\times}^{2}$ and $\mathbf{A}_{\times}^{3}$ correspond to.

UNIVERSITÄT BERN

Theoretical Tasks

Task 3

5. Apply the Floyd Transformation to the following graph.



- Use Alg 14 in the Lecture Notes
- Write down the matrix d you obtain after line 15 and the final matrix d.

Implementation Task

In this implementation task, you have to implement the shortest-path kernel and an enumerating graph kernel.

Remarks:

- The entire code must be contained within the file `PR_lecture/Exercise_5/ex5_a.py` and `PR_lecture/Exercise_5/ex5_b.py`.
- You are allowed to modify the code as much as you want, including changing function signatures, creating new functions or classes, and so on.

UNIVERSITÄT BERN

Implementation Task

Ex a - Shortest-path kernel

In this first task, your goal is to implement the shortest-path kernel presented in the lecture notes (Section 9.2).

Remarks:

- Implement the Floyd transformation presented in the lecture notes.
- Use equation 9.5 in the lecture notes for the definition of $\kappa_{\text{path}}(e_1, e_2)$

UNIVERSITÄT BERN

Implementation Task

Idea of code structure

```python
import networkx as nx
import numpy as np

from utils import load_all_graphs, draw_all_graphs


def shortest_path_kernel(s1: nx.Graph, s2: nx.Graph) -> float:
    # Code here
    pass


def main():
    pass


if __name__ == '__main__':
    main()
```

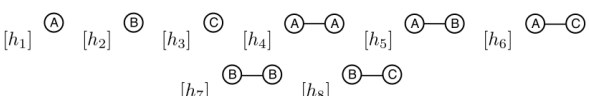
UNIVERSITÄT BERN

Implementation Task

Ex b - Enumerating kernel

In this second task, your goal is to implement an enumerating kernel. Some graph kernels are defined based on explicit enumerations of predefined substructures (cycles, trees, subgraphs, walks, etc.).

Implement the graph kernel $\kappa(g.g')$ by using the following reference graphlets h_1 to h_8.



Remarks:

- You can eventually use functions present in `networkx.algorithms`.

UNIVERSITÄT BERN

Implementation Task

Idea of code structure

```python
from typing import List

import networkx as nx
import numpy as np

from utils import load_all_graphs


def graphlet_kernel(graph1: nx.Graph, graph2: nx.Graph, graphlets: List[nx.Graph]) -> float:
    """

    Args:
        graph1:
        graph2:
        graphlets:

    Returns:

    """
    # Code Here
    pass


def main():
    #Code Here
```